Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Tugas 3 II4021 Kriptografi – Semester II tahun 2025/2026

Pembuatan Aplikasi Chat Web dengan Kriptografi Kunci-Simetri dan Kunci-Publik

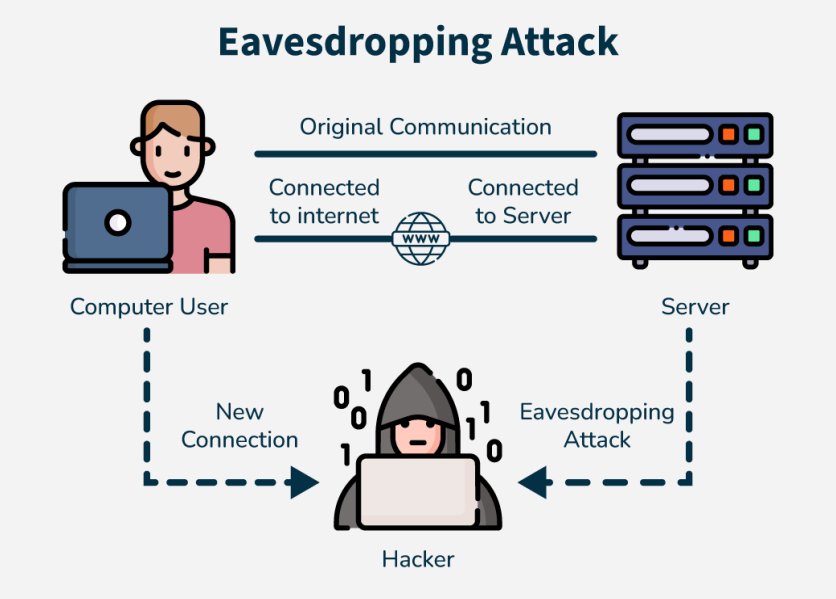
---

Batas pengumpulan	**:** Minggu, 10 Mei 2026 Pukul 23.59
Tempat pengumpulan	**:** Form Pengumpulan
Anggota kelompok	**:** 2-3 orang
Sheets kelompok	**:** Sheets Kelompok
QnA	**:** QnA Tugas II4021 Kriptografi

Revisi 1: [23/04/2026] – Memperjelas ketentuan pada bagian Perancangan dan Implementasi.

Latar Belakang

Aplikasi chat berbasis web semakin banyak digunakan karena menawarkan komunikasi yang cepat dan praktis. Namun, di balik kemudahan tersebut, terdapat risiko keamanan seperti penyadapan isi pesan, pencurian sesi, dan manipulasi pesan. Dalam kondisi seperti ini, mekanisme login saja tidak cukup, karena autentikasi pengguna tidak menjamin bahwa pesan yang dikirim akan tetap aman di jaringan.



Gambar 1. Eavesdropping Attack (GeeksforGeeks, "What is an Eavesdropping Attack?")

Oleh karena itu, diperlukan mekanisme keamanan yang tidak hanya mengatur akses pengguna, tetapi juga melindungi komunikasi antar pengguna. Pada tugas ini, aplikasi chat dirancang dengan menerapkan beberapa mekanisme kriptografi modern untuk mendukung autentikasi, pembentukan kunci komunikasi, dan enkripsi pesan.

Spesifikasi dan Ketentuan Program

A. Gambaran Umum Sistem

Aplikasi yang akan dibangun merupakan aplikasi chat berbasis web yang mengintegrasikan autentikasi, pembentukan kunci komunikasi, dan enkripsi pesan dalam satu alur sistem. Mekanisme keamanan utama yang digunakan meliputi **JSON Web Token (JWT)** untuk autentikasi, **Elliptic Curve Diffie-Hellman (ECDH)** untuk pembentukan *shared secret*, serta **Advanced Encryption Standard (AES)** untuk enkripsi dan dekripsi isi pesan.

Setelah pengguna berhasil login, sistem akan memberikan JWT yang digunakan untuk mengakses fitur aplikasi. Ketika dua pengguna mulai berkomunikasi, sistem menjalankan proses *key exchange* menggunakan ECDH untuk menghasilkan *shared secret* yang sama. Nilai tersebut kemudian diproses menggunakan fungsi derivasi kunci untuk menghasilkan kunci simetris yang digunakan oleh AES-256 untuk mengenkripsi dan mendekripsi pesan.

Dengan rancangan ini, sistem menggunakan satu kunci untuk setiap pasangan pengguna, sedangkan server hanya berperan sebagai perantara yang "buta" untuk autentikasi dan penerusan data tanpa memegang kunci komunikasi maupun mengetahui isi plainteks pesan.

B. Registrasi Pengguna

Pada tahap registrasi, pengguna membuat akun dengan memasukkan **email** dan **password** ke dalam aplikasi. Pada saat yang sama, klien membangkitkan pasangan kunci ECDH yang terdiri atas *private key* dan *public key* untuk pengguna. *Private key* terlebih dahulu dienkripsi menggunakan AES-256 dengan kunci yang diturunkan dari password pengguna, kemudian bersama *public key* dikirim ke server untuk disimpan sebagai bagian dari data pengguna. Varian AES yang digunakan dibebaskan (misalnya AES-256-CBC) dan wajib dijelaskan pada laporan.

Ketentuan pada tahap ini adalah sebagai berikut:

1. Password **tidak boleh disimpan dalam bentuk plainteks**.
2. Password wajib diolah terlebih dahulu menggunakan mekanisme *hashing* yang dikombinasikan dengan *salt* unik sebelum disimpan ke basis data.
3. Algoritma *hashing* password yang digunakan dibebaskan dan boleh memakai *library* ataupun *built-in*.

4. Mekanisme KDF yang digunakan untuk menurunkan kunci enkripsi *private key* dibebaskan.
5. Varian ECDH yang digunakan juga dibebaskan (misalnya X25519 atau varian lain yang sesuai).
6. Server menyimpan email, *hashed password*, *salt*, *public key*, dan *private key* dalam bentuk terenkripsi, serta parameter atau metadata lain yang diperlukan untuk proses pemulihan kunci.

C. Login dan Autentikasi

Pada tahap login, pengguna memasukkan email dan password ke dalam aplikasi. Sistem kemudian mencocokkan password dengan data yang tersimpan pada basis data. Apabila verifikasi berhasil, sistem akan menerbitkan JSON Web Token (JWT) yang digunakan sebagai bukti autentikasi untuk mengakses fitur-fitur aplikasi. JWT adalah string yang mewakili sekumpulan klaim dalam bentuk objek JSON. Pada tugas ini, JWT diimplementasikan dalam format JWS dan ditandatangani secara digital menggunakan ECDSA.

Ketentuan pada tahap ini adalah sebagai berikut:

1. Buatkan *library* yang mengimplementasi fungsi ***sign*** dan ***verify*** JWT dengan ketentuan pada tabel berikut:

Tabel 1. Tabel Definisi Implementasi Library

<table>
<tr><th>Fungsi</th><th>Input</th><th>Output</th></tr>
<tr><td rowspan="2">sign</td><td>header: {
alg: "ES256" | "ES384" | "ES512",
typ: "JWT",
}

Parameter header harus ada. ES256 menggunakan kurva P-256 dengan SHA-256, ES384 menggunakan kurva P-384 dengan SHA-384, dan ES512 menggunakan kurva P-521 dengan SHA-512.</td><td rowspan="2">Return JWT sebagai string dengan format base64url(header).base64url(payload).base64url(signature)

Contoh:
eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJmb28iOiJiYXIiLCJpYXQiOjE3NzY3NjgyNzMsImV4cCI6MzU1NzEzNjU0Niwi YXVkIjoiaHR0cHM6Ly9leGFtcGxlLmNvbSIsImlzcyI6ImpvZSIsInN1YiI6IjEyMzQ1Njc4OTAiLCJqdGkiOiIxMDA3YTk4Zi1kOTZmLTQ3NGYtYTlhYS0zYmY3N2NmMDczMTUifQ.4WuxXe1WcSZOqS4Qb_ZgpaW8oG8iHBEB3c7baDGA</td></tr>
<tr><td>claims: {
iss: string,
sub: string,
aud: string,
exp: NumericDate,
nbf: NumericDate,
iat: NumericDate,
jti: string,
}</td></tr>
</table>

	Seluruh parameter claims **opsional.**	0MM71BxTvViXeVXwnROkWfnkhZoOpmYVIJaGZYkDbjd7FA

	payload: {
	    foo: string | number | boolean | null | object | array,
	    bar: string | number | boolean | null | object | array,
	    ...
	}
	
	Payload bisa berisi ***registered claim***, ***public claim***, ataupun ***private claim***. Jika *key* di *payload* sama dengan *key* di parameter *claims*, gunakan nilai *key* di parameter *claims*. *Payload* harus dapat di-*serialized* menjadi JSON.	

	privateKey: PEM string	
verify	jwt: string	*Throw error* jika gagal. *Return decoded* JWT (*header + payload + signature*)
	publicKey: PEM string	

	options: {
	    algs: ("ES256" | "ES384" | "ES512")[],
	    iss: string,
	    sub: string,
	    aud: string,
	    ignoreExp: boolean,
	    ignoreNbf: boolean,
	    jti: string,
	}
	
	Seluruh parameter *options* **opsional.** Langkah validasi disesuaikan dengan RFC 7519 bagian 7.2. Jika JWT tidak valid, *throw error* dengan pesan yang sesuai. Jika ada *option* yang dispesifikasikan dan tidak sesuai dengan *decoded* JWT, *throw error* dengan pesan yang sesuai.	

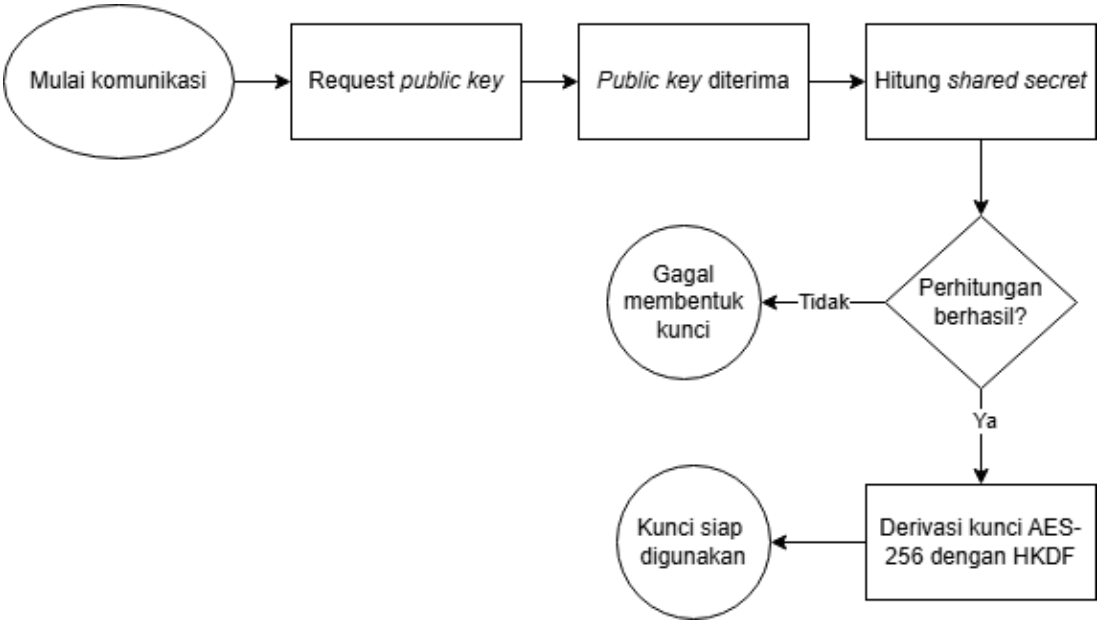
Format input dibebaskan selama memenuhi ketentuan.

2. Diperbolehkan untuk menambahkan fungsi utilitas tambahan untuk mempermudah implementasi.
3. Implementasi *library* boleh menggunakan library kriptografi *low-level* untuk alur sign dan verify (e.g., *library* **crypto** untuk JavaScript, **PyCryptodome** untuk Python).
4. Implementasi *library* terbatas hanya pada fungsionalitas *sign* dan *verify*. Boleh menggunakan library apapun untuk menyimpan JWT pada cookie.

D. Daftar Kontak

Setelah berhasil login, pengguna dapat melihat halaman daftar kontak. Halaman ini akan menampilkan semua email pengguna yang pernah mendaftarkan diri pada aplikasi, kecuali akun sendiri. Pengguna dapat memilih salah satu kontak tersebut untuk memulai percakapan.

E. Pembentukan Kunci Komunikasi dan *Key Exchange*



Gambar 2. Proses Pembentukan Kunci dan Key Exchange

Ketika dua pengguna mulai berkomunikasi, kedua sisi menjalankan proses *key exchange* menggunakan ECDH untuk menghasilkan *shared secret* yang sama. Dalam proses ini, masing-masing sisi mengambil *public key* lawan komunikasi dari server, menghitung *shared secret* dengan ECDH, memprosesnya menggunakan fungsi derivasi kunci HKDF, lalu menggunakan hasilnya sebagai kunci AES-256. Kunci simetris inilah yang digunakan untuk enkripsi dan dekripsi pesan antar kedua pengguna. Server hanya berperan dalam penyimpanan dan penerusan nilai-nilai publik serta data terenkripsi, dan **tidak memegang *shared secret* maupun kunci komunikasi**.

F. Pengiriman Pesan

Pada tahap ini, pengguna dapat mengirim pesan kepada lawan komunikasi. Isi pesan tidak dikirim dalam bentuk plainteks, melainkan harus dienkripsi terlebih dahulu menggunakan AES. Dalam implementasi ini, algoritma yang digunakan adalah AES-256. Varian AES-256 yang digunakan dibebaskan (misalnya AES-256-CBC) dan wajib dijelaskan pada laporan.

Server hanya berperan sebagai perantara yang meneruskan data terenkripsi ke penerima dan tidak mengetahui isi plainteks pesan. Pesan tidak harus *real-time* (boleh *polling*, *simple REST*, atau *real-time* menggunakan WebSocket).

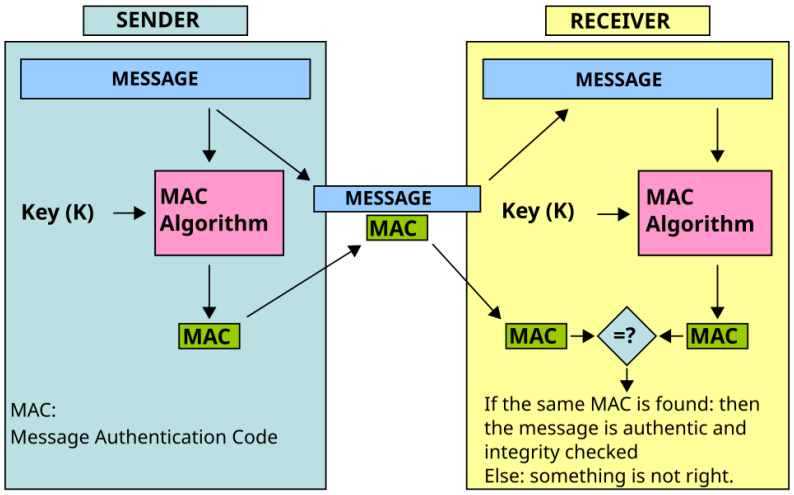
Ketentuan pada tahap ini adalah sebagai berikut:

1. Apabila proses dekripsi gagal, sistem harus menampilkan penanda bahwa pesan tidak dapat didekripsi.
2. Pesan yang berhasil didekripsi harus ditampilkan kembali dalam bentuk plainteks pada antarmuka aplikasi.
3. Format data dapat berupa JSON dengan struktur sebagai berikut. Struktur ini hanya merupakan contoh, dan penyusunan payload dibebaskan.

```
{
  "sender_email": "diero@gmail.com",
  "receiver_email": "arga@gmail.com",
  "ciphertext": "650f579e627fdef3593bd3796fc38dc5",
  "iv": "493c334b726970746f67726166692121",
  "timestamp": "2026-04-16T05:32:27.377Z"
}
```

G. Bonus

a. Integritas dan Autentikasi Pesan (5)



Gambar 3. Ilustrasi cara kerja MAC (Wikipedia, "Message authentication code")

Saat pesan dikirim, sistem juga menghitung Message Authentication Code (MAC) berdasarkan isi pesan dan kunci yang relevan, lalu mengirimkannya bersama payload. Mekanisme ini digunakan untuk membantu memastikan integritas pesan serta autentikasi pengirim.

Di sisi penerima, nilai MAC tersebut diverifikasi terlebih dahulu sebelum pesan diproses lebih lanjut. Apabila nilai MAC tidak sesuai, pesan harus ditandai sebagai tidak valid. Mekanisme implementasi MAC, termasuk algoritma, material kunci, dan cara integrasinya ke dalam payload pesan dibebaskan dan wajib dijelaskan pada laporan.

b. Unit Test untuk Library JWT (5)

Dalam proses pengembangan perangkat lunak, *library* yang baik tidak hanya dituntut berfungsi pada skenario normal, tetapi juga mampu menangani berbagai kondisi batas dan input yang tidak valid secara tepat. Oleh karena itu, implementasi library JWT pada tugas ini perlu disertai dengan unit test untuk membantu memastikan bahwa fungsi-fungsi yang dikembangkan telah bekerja sesuai spesifikasi dan cukup *robust* terhadap berbagai kemungkinan kasus.

Pada bonus ini, kalian diminta membuat unit test untuk fungsi ***sign*** dan ***verify*** pada *library* JWT yang diimplementasikan. Pengujian harus mencakup seluruh *happy path* yang relevan, serta minimal lima *edge case* untuk masing-masing fungsi. Contoh *edge case* yang dapat diuji antara lain adalah *header* yang tidak valid, algoritma yang tidak didukung, token dengan format yang tidak valid, *signature* yang tidak sesuai, *claim* yang tidak memenuhi

validasi, atau input *payload* yang tidak dapat diproses sebagaimana mestinya. *Library* atau *framework unit test* yang digunakan dibebaskan dan wajib dijelaskan pada laporan.

c. Docker (5)

Untuk mempermudah dalam menjalankan, menguji, dan mendemonstrasikan aplikasi secara konsisten di berbagai lingkungan, sistem dapat dikemas menggunakan Docker. Dengan pendekatan ini, seluruh komponen aplikasi beserta dependensinya dapat dijalankan dalam *container* sehingga mengurangi permasalahan perbedaan konfigurasi antar perangkat.

Pada bonus ini, kalian diminta menyediakan konfigurasi Docker untuk menjalankan aplikasi yang dikembangkan. Konfigurasi tersebut minimal harus memungkinkan komponen utama sistem, seperti client, server, dan layanan penyimpanan atau basis data apabila digunakan, dijalankan dengan langkah yang sederhana dan terdokumentasi. Peserta dapat menggunakan Dockerfile, docker-compose.yml, atau mekanisme lain yang relevan. Struktur *container*, pemisahan *service*, serta strategi konfigurasi dibebaskan, tetapi harus dijelaskan pada laporan dan README. Aplikasi yang dijalankan melalui Docker harus tetap memenuhi alur minimum yang telah ditentukan pada spesifikasi utama.

H. Prosedur Pengerjaan

Berikut adalah ketentuan pengerjaan tugas ini:

1. Batas pengisian kelompok di Sheets kelompok: **Jumat, 24 April 2026 pukul 23.59**
2. Implementasi aplikasi dilakukan berbasis **web**. Kakas yang digunakan untuk *client*, *server*, dan storage atau basis data **dibebaskan**.
3. Seluruh operasi kriptografi di sisi klien, termasuk **ECDH**, **HKDF**, dan **AES**, wajib menggunakan **Web Crypto API**.
4. Penggunaan library yang secara langsung menangani *secure messaging* tidak diperbolehkan.
5. Jika ada *library* yang tidak disebutkan di spesifikasi dan dianggap dapat digunakan, harap tanyakan terlebih dahulu di *sheets* QnA.
6. Program harus ditulis dengan struktur yang rapi, mudah dibaca, dan disertai komentar yang jelas pada bagian yang diperlukan.
7. Dilarang menyalin kode program secara langsung dari internet maupun menggunakan keluaran mentah (*raw output*) dari *Generative AI*.
8. Antarmuka aplikasi dibebaskan, tetapi harus tetap fungsional, mudah digunakan, dan memenuhi alur minimum yang telah ditentukan.

Pengumpulan

Pengumpulan dilakukan dengan melakukan ***release*** pada repositori Github sebelum tenggat waktu pengerjaan. Revisi pengumpulan dapat dilakukan dengan membuat release baru.

1. Repositori Perangkat Lunak

Repositori harus memuat:

a. Kode sumber (*source code*).
b. Berkas eksekutabel (*executable file*), jika relevan.
c. Dokumen laporan
d. README yang minimal berisi:
    i. Nama dan deskripsi program.
    ii. Teknologi yang digunakan (*tech stack*).
    iii. Dependensi.
    iv. Tata cara menjalankan program.
    v. *Environment/configuration* yang digunakan.

2. Video Demo

Video demo berdurasi maksimal 10 menit, berisi:

a. Deskripsi singkat program.
b. Penjelasan singkat rancangan, terutama bagian yang dibebaskan atau dirancang sendiri.
c. Demo kasus uji utama sesuai ketentuan pengujian.

Apabila terdapat proses yang memerlukan waktu lama untuk dijalankan, video demo dapat dipotong pada bagian tersebut selama alur demonstrasi tetap jelas.

3. Laporan

Laporan diunggah pada repositori dengan nama **NIM1_NIM2_NIM3_Tugas3_II4021.pdf**, berisi:

a. Foto anggota kelompok di *cover* laporan sebagai pengganti logo gajah.
b. Pernyataan tidak melakukan kecurangan yang ditandatangani dengan format sebagai berikut:

> Kami menyatakan bahwa kode program yang dihasilkan bukan merupakan hasil salinan mentah (*raw output*) dari *Generative AI*, melainkan hasil pengembangan dan penulisan mandiri.
>
> | | | |
> |---|---|---|
> | [Tanda tangan Mahasiswa 1] | [Tanda tangan Mahasiswa 2] | [Tanda tangan Mahasiswa 3] |
> | [Nama Mahasiswa 1] | [Nama Mahasiswa 2] | [Nama Mahasiswa 3] |

c. Teori singkat mengenai JWT, Elliptic Curve Diffie-Hellman, AES, dan konsep relevan lainnya (termasuk bonus).
d. Perancangan dan implementasi, berisi penjelasan rancangan dan implementasi program, **bukan penyalinan isi codebase**. Setiap bagian yang dijelaskan harus disertai nama file

yang relevan pada repository. Pastikan untuk menjelaskan implementasi bagian yang dibebaskan dan bonus.

e. Pengujian program dan analisis hasil, dengan minimal kasus uji mencakup:
    i. Uji autentikasi dan manajemen pengguna:
        1. Registrasi dengan data valid, perlihatkan bahwa akun berhasil dibuat dan data tersimpan dengan benar.
        2. Registrasi dengan email yang sudah terdaftar, perlihatkan bahwa sistem menolak input tersebut.
        3. Login dengan *password* yang benar, perlihatkan bahwa JWT berhasil diterbitkan.
        4. Login dengan *password* yang salah, perlihatkan bahwa autentikasi gagal.
    ii. Uji implementasi JSON Web Token (JWT):
        1. Gunakan fungsi *sign* untuk menghasilkan JWT, lalu lakukan *verify* dengan *public key* yang benar, perlihatkan bahwa token valid dan *payload* dapat dibaca.
        2. Lakukan *verify* dengan *public key* yang salah, perlihatkan bahwa verifikasi gagal.
        3. Uji token dengan format tidak valid, perlihatkan bahwa sistem melempar *error*.
        4. Uji token dengan klaim waktu (exp/nbf) yang tidak sesuai, perlihatkan bahwa token ditolak (kecuali opsi *ignore* digunakan).
    iii. Uji pembentukan kunci komunikasi (ECDH dan KDF):
        1. Dua pengguna melakukan *key exchange*, perlihatkan bahwa kedua sisi menghasilkan *shared secret* yang sama.
        2. Tunjukkan bahwa *shared secret* tersebut berhasil diturunkan menjadi kunci simetris yang digunakan untuk enkripsi pesan.
    iv. Uji enkripsi dan dekripsi pesan:
        1. Kirim pesan dengan kunci yang benar, perlihatkan bahwa pesan berhasil didekripsi dan ditampilkan dalam bentuk plainteks.
        2. Gunakan kunci yang salah atau data yang tidak sesuai, perlihatkan bahwa proses dekripsi gagal dan sistem menampilkan penanda *error*.
    v. (Bonus) Uji integritas dan autentikasi pesan (MAC):
        1. Kirim pesan dengan MAC yang valid, perlihatkan bahwa pesan diterima dan diproses dengan benar.
        2. Ubah sebagian isi pesan atau MAC, perlihatkan bahwa verifikasi gagal dan pesan ditandai tidak valid sebelum didekripsi.
f. Kesimpulan dari hasil implementasi,
g. Daftar pustaka,
h. Lampiran berisi:
    i. Pranala repositori

ii. Pranala video demo
iii. Pembagian tugas

Referensi

- Dokumentasi Web Crypto API
- Implementasi JWT Sesuai Standar RFC 7519
- Contoh Library JWT